# Research Report 209

## Shallow Multiplication Circuits and Wise Financial Investments

**Michael S Paterson and Uri Zwick**

**RR 209**

Paterson, Pippenger and Zwick have recently obtained a general theory that describes the optimal way in which given carry-save adders can be combined into carry-save networks. Their work produces, in particular, multiplication circuits of depth 3.71 $\log_2 n$ (these circuits put out two numbers whose sum is the result of the multiplication).
In this work an extension of the above general theory is obtained. We now consider carry-save adders that may receive inputs and produce outputs using several different representation methods. We describe the optimal way of utilising any such collection of carry-save adders. The optimality proof uses the min-max theorem of game theory.
By using several different representation standards, the depth of multiplication circuits can be surprisingly reduced to 3.48 $\log_2 n$ (again two output numbers are produced). We introduce bit level redundancy by using a novel coding scheme in which each bit is distributed over four wires. Interestingly, the information on these four wires is usually not transmitted simultaneously.
Finally, an analogy is made between the optimisation problem faced by the circuit designer and the optimisation problem faced by an investor, offered a collection of financial investment plans, each involving perhaps several different currencies. This analogy is used to obtain intuitive explanations of the results obtained.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

February 1992

# Shallow Multiplication Circuits and Wise Financial Investments *

*Michael S. Paterson*
Dept. of Computer Science
University of Warwick
Coventry CV4 7AL, England

*Uri Zwick*
Dept. of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel

**Abstract**

Paterson, Pippenger and Zwick have recently obtained a general theory that describes the optimal way in which given carry-save adders can be combined into carry-save networks. Their work produces, in particular, multiplication circuits of depth $3.71 \log_2 n$ (these circuits put out two numbers whose sum is the result of the multiplication).

In this work an extension of the above general theory is obtained. We now consider carry-save adders that may receive inputs and produce outputs using several different representation methods. We describe the optimal way of utilising any such collection of carry-save adders. The optimality proof uses the min-max theorem of game theory.

By using several different representation standards, the depth of multiplication circuits can be surprisingly reduced to $3.48 \log_2 n$ (again two output numbers are produced). We introduce bit level redundancy by using a novel coding scheme in which each bit is distributed over four wires. Interestingly, the information on these four wires is usually not transmitted simultaneously.

Finally, an analogy is made between the optimisation problem faced by the circuit designer and the optimisation problem faced by an investor, offered a collection of financial investment plans, each involving perhaps several different currencies. This analogy is used to obtain intuitive explanations of the results obtained.

## 1. Introduction

A carry-save adder (or *CSA* for short) is a unit with $k$ input numbers and $\ell$ output numbers ($\ell < k$), with the property that the sum of the outputs is always equal to the sum of the inputs.

*This work was partially supported by the ESPRIT II BRA Programme of the EC under contract # 3075 (ALCOM). Authors' e-mail addresses: msp@dcs.warwick.ac.uk and zwick@math.tau.ac.il

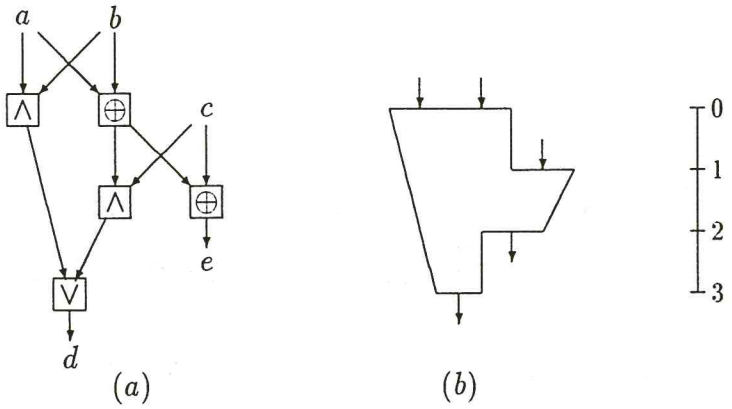


Figure 1.1. A 3-bit full-adder and the resulting $3 \rightarrow 2$ carry-save adder

It has long been known (see [2],[4],[9]) that constant depth carry-save adders can be built and that they can be used to obtain logarithmic depth circuits for multiple addition, multiplication and related problems.

Since the multiplication of two $n$-bit integers is such a basic task it is interesting to investigate the exact depth it requires. In particular we would be interested in knowing the minimal constant $\delta$ for which there exist multiplication circuits of depth $(\delta + o(1)) \log_2 n$.

An attempt to tackle this problem was made in [6],[7]. While that work did not identify this minimal $\delta$, a task which is probably still far beyond our reach, it did supply a good upper bound ($\delta < 3.71$) and presented a general theory that described the optimal way in which given carry-save adders could be used to construct carry-save networks that reduce the sum of $n$ numbers to the sum of only a fixed number of numbers. In particular, it was shown that the $3.71 \log_2 n$ depth construction was optimal within its class, i.e., within the class of networks built from $CSA_{3\rightarrow 2}$'s ($3 \rightarrow 2$ carry-save adders) made up of arrays of standard $FA_3$'s (3-bit full-adders).

In this work a generalisation of the theory developed in [6],[7] is presented. We begin the description of this generalised theory with two concrete examples.

Fig. 1.1(a) shows the standard implementation of an $FA_3$. This implementation, as all other implementations considered in this paper, uses only dyadic (2-input) gates. Using this implementation, $CSA_{3\rightarrow 2}$ units with the time characteristics shown in Fig. 1.1(b) can be constructed. The $CSA_{3\rightarrow 2}$ unit of Fig. 1.1(b) is said to have the *characteristic polynomial* $2 + x - x^2 - x^3$. Each input contributes a positive term and each output a negative term to this polynomial. The terms are all powers of $x$, where the exponent depends on the time at which this input or output is supplied or produced. It is an immediate consequence of the theory of [6],[7], that the optimal networks for the carry-save addition of $n$ numbers that could be constructed using this $CSA_{3\rightarrow 2}$ as a basic building block have depth about $\log_\lambda n$ where $\lambda \simeq 1.20557$ is the *principal root*, i.e., the largest real root greater than 1 of the characteristic polynomial of the unit. Since $\log_\lambda n \simeq 3.71 \log_2 n$, this yields the upper bound referred to earlier.

Assume that we have at our disposal the collection of $CSA_{3\to 2}$'s given in Fig. 1.2. The terms $s_0, s_1$ and $s_2$ labeling the inputs and outputs of these units, describe the *standard* in which each input is required and in which each output is produced. We do not specify for the time being, what these different standards are. We are not allowed to connect, say, an $s_2$-output directly into an $s_0$-input (this may cause a short circuit). We have however the means of translating between the different standards, if we so wish. In this case we assume that we can use the conversion units shown in Fig. 1.3. Some of these translations are done instantaneously (such as translating an $s_0$ to an $s_1$ or $s_2$) but others take time.

We are now given $n$ input numbers, in the $s_0$ standard, and we want to build a network that will return two numbers, again in the $s_0$ standard whose sum is equal to the sum of the $n$ original numbers. What is the minimal depth that such a network would require?

The theory we develop here gives a general answer to questions of this type. Given any collection of units with possibly many different input and output standards, we describe the way in which they could be combined optimally to reduce the sum of $n$ inputs to the sum of only a fixed number of outputs. We will assume that conversion between any two standards is possible, although such a conversion may sometimes incur a large delay. Nonetheless, the possibility of such conversions shows that the standard in which the initial inputs and final outputs are required is of little importance, as the initial and final conversions will only add a fixed delay to the networks. The fixed number of numbers obtained as the output of these networks may be taken to be the minimal number of outputs among the *CSA* units available.

An application of this general theory to the specific question posed earlier concerning the collections given in Fig. 1.2 and Fig. 1.3 will give an answer of about $3.48 \log_2 n$.

As the reader may have suspected, the collections of Fig. 1.2 and Fig. 1.3 were not chosen arbitrarily. We have found three transmission standards $s_0$, $s_1$ and $s_2$ for which each of the units shown there is realisable using 2-input binary gates. Thus the depth $3.48 \log_2 n$ obtained does correspond to a real construction.

In the next section we describe the three transmission standards $s_0$, $s_1$ and $s_2$ and describe the construction of the units shown in Fig. 1.2 and Fig. 1.3.

In Section 3 we present the analogy between the network construction problem and a financial investment problem. This analogy intuitively suggests the idea of introducing the *exchange rates* that support the proof of the lower bound in Section 4 and that of introducing the *portfolios* in Section 6.

To get optimal constructions that match the lower bounds of Section 4, we have to introduce a reasonable *causality* assumption that states that an output of a unit can only depend on inputs to that unit supplied at earlier times. The causality assumption and its consequences are considered in more detail in Section 5.

Using the causality assumption, we present in Section 6 constructions that match the lower bounds of Section 4. The proof that the lower and upper bounds of Sections 4 and 6 do in fact match uses the min-max theorem of game theory and it is given in Section 7.

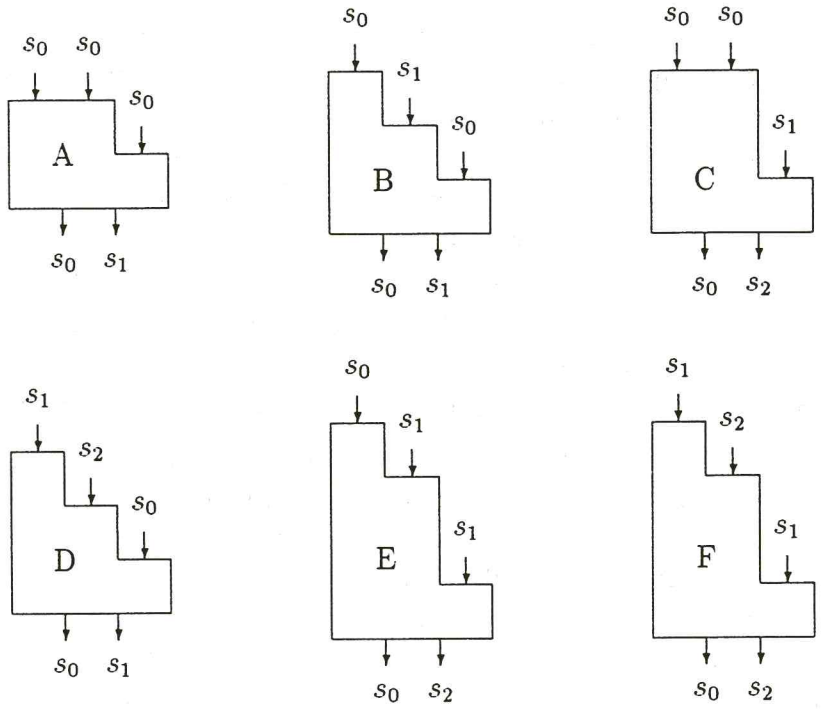


Figure 1.2. A collection of *CSA* units that use three different standards

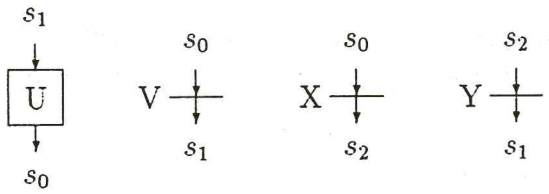


Figure 1.3. Conversion units between the three different standards

We end the extended abstract by some concluding remarks, open problems and suggestions for future work.

# 2. Split-input full adders

Consider the standard implementation of an $FA_3$ given in Fig. 1.1(a). Note that a delay is introduced on the left input wire to the OR gate computing the carry output $d$. This seems to suggest that the design could be improved, but how?

The answer is simple. We do not have to insist on producing the output $d$ as a single entity. We can simply remove the bottom OR gate and get two output bits $d_1^{\vee}, d_2^{\vee}$ such that $d = d_1^{\vee} \vee d_2^{\vee} = d_1^{\vee} \oplus d_2^{\vee}$ (note that $d_1^{\vee} = a \wedge b$ and $d_2^{\vee} = (a \oplus b) \wedge c$ are never 1 simultaneously). If we are then required to OR (or XOR) $d$ with another bit $f$ supplied at

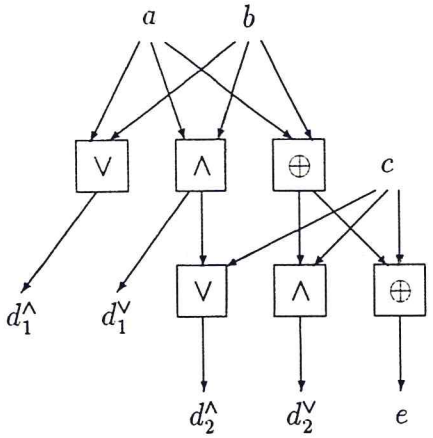


Figure 2.1. An implementation of unit $A$

the time $d_1^\vee$ is produced, we can reassociate $(d_1^\vee \vee d_2^\vee) \vee f$ as $(d_1^\vee \vee f) \vee d_2^\vee$, thereby gaining one time unit. It is the accumulation of these time units that will enable us to lower the delay of the carry-save addition and multiplication circuits.

But what will we do when required to AND $d$ with another bit? It seems that we would then have to OR the two fractions $d_1^\vee, d_2^\vee$, losing the time unit we tried to gain. To overcome this problem we note that we can easily get, within the same time limits used to obtain the pair $d_1^\vee, d_2^\vee$, another pair $d_1^\wedge, d_2^\wedge$ such that $d = d_1^\wedge \wedge d_2^\wedge$. To get this pair we simply use the following alternative majority formula $d = (a \vee b) \wedge ((a \wedge b) \vee c)$. We may therefore take $d_1^\wedge = a \vee b$ and $d_2^\wedge = (a \wedge b) \vee c$.

A description of the unit we get, which we call unit $A$ is given in Fig. 2.1. Note that the output bit $d$ is coded over four wires $(d_1^\vee, d_1^\wedge, d_2^\vee, d_2^\wedge)$ and that the information on the wires $d_2^\vee, d_2^\wedge$ flows one unit of time after the information on the wires $d_1^\vee, d_1^\wedge$.

We consider normal input and output bits to conform to standard $s_0$, and split bits, like $d$ of the previous paragraphs, to conform to standard $s_1$. We consider a split bit $(d_1^\vee, d_1^\wedge, d_2^\vee, d_2^\wedge)$ to be produced (or supplied) at time $t$ if $d_1^\vee, d_1^\wedge$ are produced (or supplied) at time $t-1$ and $d_2^\vee, d_2^\wedge$ are produced (or supplied) at time $t$. We will also encounter split bits of standard $s_2$ in which the time difference between the availability of the two pairs $d_1^\vee, d_1^\wedge$ and $d_2^\vee, d_2^\wedge$ is two time units.

With these conventions we see that the time characteristics of unit $A$ of Fig. 2.1 do correspond to those of unit $A$ of Fig. 1.2.

The output bits $d_2^\vee = (a \oplus b) \wedge c$, $d_2^\wedge = (a \wedge b) \vee c$ and $e = (a \oplus b) \oplus c$ were all obtained using formulae with the generic form $(a \circ b) \circ c$. Other generic forms that may be used to design split-bit $FA_3$ units with no internal delay are given in Table 1. To get concrete formulae for $d_2^\vee, d_2^\wedge$ and $e$ from these generic forms, the generic connective $\circ$ should be replaced by an appropriate sequence of $\vee, \wedge$ and $\oplus$ connectives. To get the concrete formulae for $e$ for example, all the connectives should be taken to be $\oplus$ (and $a_1, a_2, b_1, b_2, c_1, c_2$ should be taken to be $a_1^\vee, a_2^\vee, b_1^\vee, b_2^\vee, c_1^\vee, c_2^\vee$). The generic expressions for the fractions $d_1^\vee, d_1^\wedge$ are

$$
\begin{aligned}
A &: (a \circ b) \circ c \\
B &: (((a \circ b_1) \circ b_2) \circ c \\
C &: (((a \circ b) \circ c_1) \circ c_2 \\
D &: ((((a_1 \circ b_1) \circ a_2) \circ b_2) \circ c \\
E &: ((((a \circ b_1) \circ b_2) \circ c_1) \circ c_2 \\
F &: (((((a_1 \circ b_1) \circ a_2) \circ b_2) \circ c_1) \circ c_2
\end{aligned}
$$

Table 1. Generic forms of formulae used to compute the outputs $d_2^{\vee}, d_2^{\wedge}, e$.

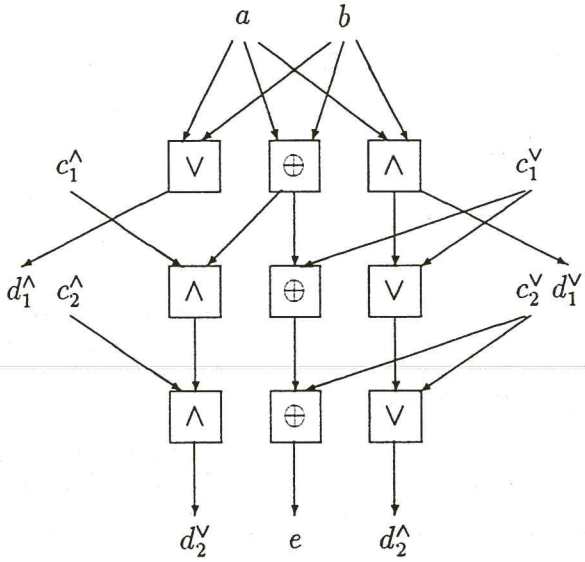


Figure 2.2. An implementation of unit $C$

obtained from those of Table 1 by removing the last level (or two levels) of nesting involving $c$ (or $c_1$ and $c_2$).

A detailed construction of units $C$ and $D$ is given in Figs. 2.2 and 2.3. The construction of units $B, E$ and $F$ goes along the same lines and requires no new ideas.

The conversion units $U, V, X, Y$ are also easily implemented. Note that we may convert an $s_0$-bit $a$ into a split bit $b = (b_1^{\vee}, b_1^{\wedge}, b_2^{\vee}, b_2^{\wedge})$ in standard $s_1$ or $s_2$ instantaneously by taking $b_2^{\vee} = b_2^{\wedge} = a$ and by letting $b_1^{\vee} = 0$ and $b_1^{\wedge} = 1$. This takes care of units $V$ and $X$. Units $U$ and $Y$ are simple to design.

The question now is whether all this extra effort was worthwhile. Can we use the collection of Fig. 1.2 to improve the result obtained using the 'gadget' of Fig. 1.1? As indicated, the answer is *yes*. Using the new collection we can design circuits with depth $3.48 \log_2 n$. This will follow from the general construction of Section 6. The results of Section 4 will show that this is indeed the best that we can squeeze from this collection.

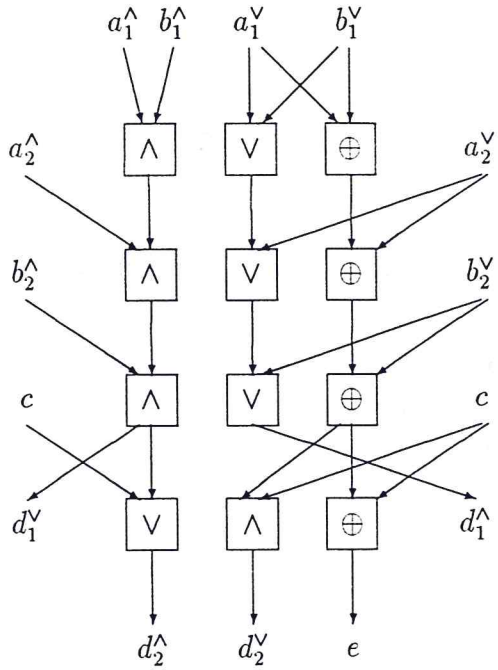


Figure 2.3. An implementation of unit $D$

# 3. Financial Plans

Before proceeding with the lower and upper bounds of Sections 4 and 5 we introduce a new interpretation to the problem with which we are faced.

Consider again the 'gadget' of Fig. 1.1(b), this time upside-down with inputs and outputs interchanged. It now corresponds to a financial plan of the following kind: invest \$1 now, \$1 in a month's time and get in return \$1 in two months' time and \$2 in three months' time. How much money could a shrewd investor with initial capital of \$2 accumulate after $n$ months?

The theory obtained in [6],[7] supplies an answer to this question. The investor can get about $c{\cdot}\lambda^n$ dollars after $n$ months, where $c$ is some fixed constant and $\lambda$ is the principal root of the characteristic polynomial of the gadget/plan. Note that this principal root simply corresponds to the effective *interest rate* of the plan.

In this work we are faced with a more complicated situation. We are given a collection of financial plans that may involve several different currencies. Gadget $C$ of Fig. 1.2, for example, corresponds to a plan that calls for the investment of \$1 and NIS 1 (new Israeli Shekel) at time 0, and promises in return £1 in a month's time and \$2 in three months' time.

# 4. Lower bounds

We first extend the definition of characteristic polynomials to the case in which more than one standard is present. An input (output) of standard $s_j$ required (produced) at time $i$ contributes the term $+s_j x^i$ ($-s_j x^i$) to such a characteristic polynomial. The symbols $s_j$ are regarded as indeterminates.

The characteristic polynomials of the units shown in Figs. 1.2 and 1.3 are therefore:

$$\begin{aligned} A &= (2+x-x^2)\cdot s_0 - x^2\cdot s_1 \\ B &= (1+x^2-x^3)\cdot s_0 + (x-x^3)\cdot s_1 \\ C &= (2-x^3)\cdot s_0 + x^2\cdot s_1 - x^3\cdot s_2 \\ D &= (x^2-x^3)\cdot s_0 + (1-x^3)\cdot s_1 + x\cdot s_2 \\ E &= (1-x^4)\cdot s_0 + (x+x^3)\cdot s_1 - x^4\cdot s_2 \\ F &= -x^4\cdot s_0 + (1+x^3)\cdot s_1 + (x-x^4)\cdot s_2 \end{aligned}$$

and

$$\begin{aligned} U &= -x\cdot s_0 + s_1 \\ V &= s_0 - s_1 \\ X &= s_0 - s_2 \\ Y &= s_2 - s_1 \end{aligned}$$

When connecting several gadgets to form a network, delays may be introduced, i.e., some outputs may be ready a few time units before they are actually needed. We may transform a network with internal delays into one without such delays by introducing explicit delay units. For simplicity we will assume that units capable of delaying an input of any given standard by one time unit are always available. Note that in the concrete example of Figs. 1.2 and 1.3, delay units for every standard are easily obtained by combining two or more converters.

The essential parameters of a network (i.e., the number of inputs and outputs it has at each time unit) are fully described by specifying the number of units of each type initiated at each time unit. Thus the characteristic polynomial of any network constructed from units $G_1, \ldots, G_m$ with characteristic polynomials $g_1(x;\mathbf{s}), \ldots, g_m(x;\mathbf{s})$ (where $\mathbf{s}$ denotes the vector of standards) could be written in the form $f_1(x)g_1(x;\mathbf{s}) + f_2(x)g_2(x;\mathbf{s}) + \ldots + f_m(x)g_m(x;\mathbf{s})$ where each $f_i(x)$ is a univariate polynomial in $x$ with non-negative integer coefficients that specifies how many copies of unit $G_i$ should be placed at each time unit.

The characteristic polynomial of a network

$$N(x;\mathbf{s}) = \sum_{i=1}^{m} f_i(x)g_i(x;\mathbf{s})$$

specifies how many inputs or outputs of each standard the network has at each time unit. We are interested in networks that take at least $n$ inputs at time 0 and return at most some fixed number $\ell$ of outputs. In this section we derive a lower bound on the total delay of such networks.

The *characteristic matrix* $M$ of a collection $G_1, \ldots, G_m$ of gadgets that use standards $s_0, \ldots, s_{k-1}$ is an $m \times k$ matrix in which the entry $m_{i,j}$ is the coefficient of the indeterminate $s_j$ in $g_i(x; \mathbf{s})$. The characteristic matrix of the collection of gadgets of Fig. 1.2 and Fig. 1.3, for example, is

$$M(x) = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ U \\ V \\ X \\ Y \end{array} \begin{array}{c} \begin{array}{ccc} s_0 & s_1 & s_2 \end{array} \\ \begin{pmatrix} 2+x-x^2 & -x^2 & 0 \\ 1+x^2-x^3 & x-x^3 & 0 \\ 2-x^3 & x^2 & -x^3 \\ x^2-x^3 & 1-x^3 & x \\ 1-x^4 & x+x^3 & -x^4 \\ -x^4 & 1+x^3 & x-x^4 \\ -x & 1 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \end{pmatrix} \end{array} .$$

We are now ready to state the lower bound.

**Theorem 1** *If $\lambda > 1$, $\mathbf{v} \in (\mathbb{R}^+)^k$, $\mathbf{v} \neq 0$ (i.e., $\mathbf{v}$ is a non-zero $k$-dimensional vector of non-negative real numbers), and $M(\lambda) \cdot \mathbf{v} \leq 0$ then any network composed of the units $G_1, \ldots, G_m$ that takes at least $n$ $s_0$-inputs and produces at most $\ell$ $s_0$-outputs (and no outputs of any other standard) has delay at least $\log_\lambda(n/\ell)$.*

**Proof :** The condition $M(\lambda) \cdot \mathbf{v} \leq 0$ implies that $g_i(\lambda; \mathbf{v}) \leq 0$ for $1 \leq i \leq m$. Let $N$ be a network with characteristic polynomial

$$N(x; \mathbf{s}) = \sum_{i=1}^{m} f_i(x) g_i(x; \mathbf{s})$$

where each $f_i(x)$ has non-negative (integral) coefficients. Since $g_i(\lambda; \mathbf{v}) \leq 0$ for $1 \leq i \leq m$, we have $N(\lambda; \mathbf{v}) \leq 0$.

By introducing delays if necessary, we may assume that all the inputs to the network $N$ are required at time 0. If $N$ has $n' \geq n$ $s_0$-inputs at time 0 and only $\ell' \leq \ell$ $s_0$-outputs at times $d_1 \leq \ldots \leq d_{\ell'}$ (and no other outputs), then

$$N(x; \mathbf{s}) = (n' - \sum_{i=1}^{\ell'} x^{d_i}) \cdot s_0 .$$

Since $N(\lambda; \mathbf{v}) \leq 0$ we get that $(n - \sum_{i=1}^{\ell'} \lambda^{d_i}) \cdot v_0 \leq 0$. The existence of converters between any two different standards implies that $v_j > 0$ for every $0 \leq j \leq k-1$, since if $v_i > 0$ but $v_j \leq 0$ then the condition $M(\lambda) \cdot \mathbf{v} \leq 0$ would be violated in a row corresponding to a converter from $s_i$ to $s_j$. In particular $v_0 > 0$ and therefore $n - \sum_{i=1}^{\ell'} \lambda^{d_i} \leq 0$, which implies that $d \geq \log_\lambda(n/\ell)$, where $d = d_{\ell'}$ is the delay of the network. □

When $M$ is the characteristic matrix for Figs. 1.2 and 1.3, the minimal $\lambda$, for which there exists a non-zero vector $\mathbf{v} \in (\mathbb{R}^+)^3$ that satisfies $M(\lambda)\cdot\mathbf{v} \leq 0$, turns out to be the principal root of the equation $3\lambda^3+2\lambda^2-2\lambda-6=0$. A corresponding vector is $\mathbf{v}=(1,v_1,v_2)$ where $v_1=-1+\lambda^{-1}+2\lambda^{-2}$ and $v_2=-1-\lambda^{-1}+\lambda^{-2}+4\lambda^{-3}$. Thus $\lambda \simeq 1.22096$, $v_1 \simeq 1.16064$, $v_2 \simeq 1.04941$ and $\log_\lambda n \simeq 3.47201 \log_2 n$. These values are obtained by solving the three simultaneous equations $A(\lambda;\mathbf{v})=C(\lambda;\mathbf{v})=D(\lambda;\mathbf{v})=0$.

Note that assigning the numerical values $v_0,\ldots,v_{k-1}$ to the indeterminates $s_0,\ldots,s_{k-1}$ corresponds to setting a fixed *exchange rate* between the different "currencies".

# 5. Causality

To get constructions that match the lower bounds of the previous section, we need to assume that the gadgets with which we are dealing are causal.

**Definition** A unit $G$ is said to be *causal* if an output produced at time $t$ depends only on inputs supplied at times strictly less than $t$.

As a consequence we get that a causal carry-save adder can produce a useful, non-constant, output only after receiving at least one input and that it cannot receive any input after all the outputs have been produced.

A further consequence of causality that will be used in the next section is the following property: if all the inputs supplied to a causal unit before time $t$ are zero, then all the outputs at or before time $t$ are constants.

Note that any gadget implemented using Boolean gates or any other conceivable technology will certainly be causal, since it cannot predict its future inputs.

The conversion units $V, X$ and $Y$ are not strictly causal as they produce their outputs instantaneously. Any combination of these units with other strictly causal units will yield a strictly causal construct however, so no problems arise in the constructions and proofs. We note that at the implementation level these units seem to violate causality in a stronger sense. Unit $V$ for example puts out a part of the $s_1$-output before it gets its $s_0$-input. However this part is identically constant.

If $G$ is a gadget, we denote by $G^{[t]}$ the gadget obtained by removing from $G$ all the inputs before time $t$ and all the outputs at or before time $t$. Note that if $G$ is causal, then $G^{[t]}$ could be emulated by feeding zeros to all the inputs required before time $t$ and ignoring all the outputs that are produced at or before time $t$ (as they are all constant anyway).

Causality has an interesting interpretation from the reversed point of view relating to financial plans. It corresponds to allowing the investor to stop all further payments to a financial plan at the cost of receiving no further returns in the future, but without any other penalty.

# 6. Constructions

We can now present our constructions.

**Theorem 2** *If $\lambda > 1$ and $\mathbf{u} \in (\mathbb{R}^+)^m$, $\mathbf{u} \neq 0$ and $\mathbf{u}^T \cdot M(\lambda) \geq 0$, then there exist networks with at least $\Omega(n)$ inputs at time 0 and at most $O(1)$ outputs, all produced before time $\log_\lambda n + O(1)$.*

**Proof :**

Let us at first ignore problems of integrality and consider "networks" that may use a non-integral number of gadgets at any level.

Let $P_{n,\lambda}(x) = n \cdot \sum_{\ell=0}^{\lceil \log_\lambda n \rceil} \lambda^{-\ell} x^\ell$, and let the "gadget" $H$ correspond to a mixed investment portfolio in the gadgets $G_1, \ldots, G_m$, with corresponding weights $u_1, \ldots, u_m$, so that $H$ has the characteristic polynomial $h(x; \mathbf{s}) = \sum_{i=1}^m u_i g_i(x; \mathbf{s})$. We consider the "network" $N_n$ with characteristic polynomial

$$N_n(x; \mathbf{s}) = P_{n,\lambda}(x) \cdot h(x; \mathbf{s})$$

built from copies of $H$ as specified by the polynomial $P_{n,\lambda}(x)$ (the coefficients of $P_{n,\lambda}(x)$ prescribe how many copies of $H$ are placed at each level).

We can classify the inputs and outputs to the "gadget" $H$ according to their standard:

$$h(x; \mathbf{s}) = \sum_{j=0}^{k-1} h_j(x) \cdot s_j \quad .$$

The condition $\mathbf{u}^T \cdot M(\lambda) \geq 0$ is equivalent to the condition $h_j(\lambda) \geq 0$ for $0 \leq j \leq k-1$. We divide each $h_j(x)$ by $1 - \lambda^{-1}x$ and get that

$$h_j(x) = (1 - \lambda^{-1}x) h'_j(x) + h''_j$$

where $h''_j \geq 0$ for $0 \leq j \leq k-1$.

Note that

$$P_{n,\lambda}(x) = n \cdot \frac{1 - \lambda^{-d} x^d}{1 - \lambda^{-1} x}$$

where $d = \lceil \log_\lambda n \rceil + 1$. Thus

$$\begin{aligned} & N_n(x; \mathbf{s}) \\ = \quad & n \cdot \frac{1 - \lambda^{-d} x^d}{1 - \lambda^{-1} x} \cdot h(x; \mathbf{s}) \\ = \quad & n \cdot \left[ (1 - \lambda^{-d} x^d) h'(x; \mathbf{s}) + \frac{1 - \lambda^{-d} x^d}{1 - \lambda^{-1} x} h''(\mathbf{s}) \right] \end{aligned}$$

where

$$h'(x; \mathbf{s}) = \sum_{j=0}^{k-1} h'_j(x) s_j \text{ and } h''(\mathbf{s}) = \sum_{j=0}^{k-1} h''_j s_j \, .$$

All the coefficients of $h''$ are non-negative and therefore correspond to inputs. As $n\lambda^{-d} < 1$ the term $-n\lambda^{-d} x^d h'(x; \mathbf{s})$ specifies a finite (perhaps fractional) number of inputs and outputs all within time $d + \partial h'$ where $\partial h'$ is the degree of $x$ in $h'(x; \mathbf{s})$.

We are left with the term $n \cdot h'(x; \mathbf{s})$. Each of the units $G_1, \ldots, G_m$ is causal, and therefore the unit $H$ obtained from them is also causal. In particular, if we let $\Delta = \partial h$, then at time $\Delta$ the unit $H$ produces some outputs and does not take in any inputs. As a consequence, all the terms in $h'(x; \mathbf{s})$ with degree $\partial h' = \Delta - 1$ in $x$ have positive signs.

Since $H$ is causal, any "network", such as $N_n$, composed from it is again causal. We now consider the network $N_n^{[\Delta-1]}$ obtained by feeding zeros to all the inputs required before time $\Delta - 1$ and ignoring all outputs at or before time $\Delta - 1$. This network has $\Omega(n)$ inputs at time $\Delta - 1$ and perhaps some additional inputs at other times, but only $O(1)$ outputs, all within time $\log_\lambda n + O(1)$. This "network" satisfies all our requirements except that it usually involves fractional gadgets.

The solution to the integrality problem is quite simple. Consider the network

$$N_n^*(x; \mathbf{s}) = \sum_{i=1}^{m} \lceil u_i P_{n,\lambda}(x) \rceil g_i(x; \mathbf{s})$$

where $\lceil \sum_{\ell=1}^{q} a_\ell x^\ell \rceil = \sum_{\ell=1}^{q} \lceil a_\ell \rceil x^\ell$.

It is easy to check that the coefficients in $N_n^*$ differ from the corresponding coefficients in $N_n$ by at most an additive constant. Thus the network $N_n^*$ will still have $\Omega(n)$ inputs at the start (time $\Delta - 1$) but may now have $O(1)$ additional outputs at every time unit up to $\log_\lambda n + O(1)$. Since these additional outputs trickle out very slowly (only a fixed number at each time unit), it is easy to reduce their sum to the sum of a fixed number of outputs at time $\log_\lambda n + O(1)$ by adding a fixed number of carry-save adders at each level. This establishes the validity of the theorem. □

Let $\lambda \simeq 1.22096$ be the principal root of the equation $3\lambda^3 + 2\lambda^2 - 2\lambda - 6 = 0$. It can be verified that

$$(2 - \lambda^3) \cdot A(\lambda; \mathbf{s}) + C(\lambda; \mathbf{s}) + \lambda^2 \cdot D(x; \mathbf{s}) = 0$$

where $2 - \lambda^3 \simeq 0.17985$. This corresponds to a portfolio using which networks of depth $\log_\lambda n \simeq 3.47201 \log_2 n$ could be constructed. These networks make optimal use of the gadgets shown in Figs. 1.2 and 1.3. Note that units $B, E, F$ are not used at all by these networks and that the converters $U, V, X, Y$ are needed (if at all) primarily at the initial and final stages.

# 7. The min-max theorem

An easy consequence of the von Neumann/Morgenstern min-max theorem of game theory (see [5]) is the following:

**Theorem 3** *For every $m \times k$ matrix $M$ of real numbers, there exists a vector $\mathbf{v} \in (\mathbb{R}^+)^k . \mathbf{v} \neq 0$, such that $M \cdot \mathbf{v} \leq 0$ or a vector $\mathbf{u} \in (\mathbb{R}^+)^m, \mathbf{u} \neq 0$, such that $\mathbf{u}^T \cdot M \geq 0$.*

This theorem establishes the optimality of the lower bounds and constructions presented in Sections 4 and 6. For every $\lambda > 1$ we either get a lower bound of $\log_\lambda n - O(1)$ or an upper

bound of $\log_\lambda n + O(1)$, or both. Continuity considerations show that, for the maximum $\lambda$ for which we get a $\log_\lambda n + O(1)$ upper bound, we also get a corresponding $\log_\lambda n - O(1)$ lower bound.

# 8. Concluding remarks

The multipliers of many present day CPU units already follow Wallace's suggestion (see [9]) and use networks of carry-save adders. Although our results are presented in the idealised Boolean circuits model, we think that some of the ideas used here may have practical value in the future.

The circuits described in this paper use the basis $\{\vee, \wedge, \oplus\}$. In [8], a construction of multipliers with depth about $4.95 \log_2 n$ over the basis $\{\vee, \wedge, \neg\}$ is presented. This construction uses as a building block a unit with 11 inputs and 4 outputs. A unit with 6 inputs and 3 outputs over the basis $\{\wedge, \oplus\}$, yielding multiplers with depth about $3.57 \log_2 n$, may also be found there. It would be interesting to know whether the methods described in the present paper yield further improvements when applied to such larger units.

From the theoretical point of view many questions remain unanswered. Is the depth of multiplication equal to the depth of multiple addition? Is there a finite collection of gadgets using which optimal depth circuits for multiple addition can be obtained?

How far from optimal are our circuits? The best lower bound currently known for the depth of multiplication over the full binary basis $B_2$ is $\log_2 n + \Omega(\log \log n)$ (see [1]). Over the basis $U_2 = B_2 - \{\oplus, \equiv\}$, the *unate* dyadic gates, a $2 \log_2 n$ lower bound on the depth of multiplication follows from Khrapchenko's lower bound [3] for formula size. Obtaining better lower bounds for multiplication is a challenging problem.


# Acknowledgement

The authors would like to thank Noga Alon for helpful discussions on the min-max theorem of game theory.


# References


[1] Fischer M.J., Meyer A.R., Paterson M.S., "$\Omega(n \log n)$ lower bounds on length of Boolean formulas," *SIAM J. Comput.*, Vol. 11 (1982), pp. 416-427.

[2] Karatsuba A., Ofman Y., "Multiplication of multidigit numbers on automata," *Soviet Physics Dokl.*, Vol. 7 (1963), pp. 595-596.

[3] Khrapchenko V.M., "A method of determining lower bounds for the complexity of $\pi$-schemes," *Mat. Zametki*, Vol. 10 (1972), pp. 83-92 (in Russian). *Math. Notes Acad.*

*Sciences USSR*, Vol. 10 (1972), pp. 474-479 (English translation).

[4] Khrapchenko V.M., "Some bounds for the time of multiplication," *Problemy Kibernet.*, Vol. 33 (1978), pp. 221-227 (in Russian).

[5] von Neumann J., Morgenstern O., *Theory of Games and Economic Behavior*, Princeton Univ. Press, 1944.

[6] Paterson M.S., Pippenger N. and Zwick U., "Faster circuits and shorter formulae for multiple addition, multiplication and symmetric Boolean functions," *Proceedings of the 31st Ann. IEEE Symp. on Found. of Comp. Sci.*, St. Louis 1990, pp. 642-650.

[7] Paterson M.S., Pippenger N. and Zwick U., "Optimal carry save networks," *Boolean function complexity: Selected papers from the LMS symposium, Durham 1990.* To appear, Cambridge Univ. Press, 1992.

[8] Paterson M.S., Zwick U., "Shallow multiplication circuits," *Proceedings of ARITH-10, the 10th IEEE Symposium on Computer Arithmetic*, Grenoble, France, 1991, pp. 28-34.

[9] Wallace C.S., "A suggestion for a fast multiplier," *IEEE Trans. Electronic Comp.* EC-13 (1964) pp. 14-17.